3. Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX) a. Push an Element on to Stack b. Pop an Element from Stack c. Demonstrate how Stack can be used to check Palindrome d. Demonstrate Overflow and Underflow situations on Stack e. Display the status of Stack f. Exit Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define MAX 4

bool fnStkFull(int);
bool fnStkEmpty(int);
void fnPush(int [], int, int*);
int fnPop(int [], int*);
void fnDisplay(int[], int);
int fnPeek(int [], int);
bool fnChkPalindrome(int);

int main(void)
{
int stkArray[MAX];
int top = -1;
int iElem, iChoice;
for(;;)
{
        printf("\nSTACK OPERATIONS\n");
        printf("====================");
        printf("\n1.Push\n
                2.Pop\n
                3.Display\n
                4.Peek\n
                5.CheckPalindrome\n
                6.DemonstarteOverflow\n
                7.Demonstarte Underflow\n
                8.EXIT\n");
        printf("Enter your choice\n");
        scanf("%d",&iChoice);
        switch(iChoice)
        {
        case 1: if(!fnStkFull(top))
  {
    printf("\nEnter element to be pushed onto the stack\n");
    scanf("%d", &iElem);
    fnPush(stkArray, iElem, &top);
  }
  else
  {
    printf("\nStack Overflow\n");
  }
break;

        case 2: if(!fnStkEmpty(top))
  {
    iElem = fnPop(stkArray, &top);
    printf("\nPopped Element is %d\n", iElem);
  }
  else
  {
    printf("\nStack Underflow\n");
  }
```

```
break;

        case 3: if(fnStkEmpty(top))
  {
     printf("\nStack Empty\n");
  }
  else
  {
     fnDisplay(stkArray, top);
  }
                                        break;

        case 4: if(!fnStkEmpty(top))
                                        {
                                                iElem = fnPeek(stkArray, top);
                                                printf("\nElement at the  top of the stack is %d\n", iElem);
                                        }
                                        else
                                                printf("\nEmpty Stack\n");
                                        break;

        case 5: printf("\nEnter number to be checked for a palindrome : ");
  scanf("%d", &iElem);
  if(fnChkPalindrome(iElem))
  {
     printf("\n%d is a palindrome\n", iElem);
  }
  else
  {
     printf("\n%d is not a palindrome\n", iElem);
  }
  break;

        case 6: if(!fnStkFull(top))
     printf("\nThere are currently %d elements in Stack\nPush %d elemnts for Stack to overflow", top+1, MAX -
(top+1));
  while(!fnStkFull(top))
  {
     printf("\nEnter an element : ");
     scanf("%d", &iElem);
     fnPush(stkArray, iElem, &top);
  }
  printf("\nStack Overflow cannot push elements onto the stack\n");
  break;

        case 7: if(!fnStkEmpty(top))
     printf("\nThere are currently %d elements in Stack\nPop out %d elemnts for Stack to Underflow", top+1, MAX -
(top+1));
  while(!fnStkEmpty(top))
  {
     iElem = fnPop(stkArray, &top);
     printf("\nPopped Element is %d\n", iElem);
  }
  printf("\nStack Underflow cannot pop elements from the stack\n");
  break;

        case 8: exit(1);

                default: printf("\nWrong choice\n");
        }
}
return 0;
```

```
}

bool fnStkFull(int t)
{
        return ((t == MAX-1) ? true : false);
}

bool fnStkEmpty(int t)
{
        return ((t == -1) ? true : false);
}

void fnPush(int stk[], int iElem, int *t)
{
        *t = *t + 1;
        stk[*t] = iElem;
}

int fnPop(int stk[], int *t)
{
        int iElem;
        iElem = stk[*t];
        *t = *t - 1;

        return iElem;
}

void fnDisplay(int stk[], int t)
{
        int i;

        printf("\nStack Contents are: \n");
        for(i = t ; i > -1; --i)
        {
                printf("\t%d\n", stk[i]);
        }
        printf("Stack has %d elements\n", t+1);
}

int fnPeek(int stk[], int t)
{
        return stk[t];
}

bool fnChkPalindrome(int iVal)
{
   int palStk[10];
   int t = -1, iDig, iRev = 0;

   int iCopy = iVal;

   while(iCopy != 0)
   {
      iDig = iCopy % 10;
      fnPush(palStk, iDig, &t);
      iCopy /= 10;
   }
   int p = 0;
   while(p <= t)
   {
      iDig = palStk[p];
      iRev = iRev *10 + iDig;
```

```
        p++;
    }
    if(iRev == iVal)
        return true;
    else
        return false;
}
```